Dialog DataStar

options | logoff | feedback | help

databases | easy search

Advanced Search:

Inspec - 1898 to date (INZZ)

limit

Search history:

No.	Database	Search term	Info added since	Results	
1	INZZ	buffer NEAR cach$ WITH (extent OR extents) WITH database$	unrestricted	0	-
2	INZZ	database$	unrestricted	136246	show titles
3	INZZ	2 AND buffer NEAR cach$	unrestricted	36	show titles
4	INZZ	3 AND (extent OR extents)	unrestricted	0	-
5	INZZ	buffer NEAR cach$	unrestricted	374	show titles
6	INZZ	5 AND (extent OR extents)	unrestricted	2	show titles
7	INZZ	6 AND (page OR pages)	unrestricted	0	-
8	INZZ	5 AND (page OR pages)	unrestricted	39	show titles
9	INZZ	8 AND database$	unrestricted	10	show titles
10	INZZ	9 AND table$	unrestricted	0	-

hide | delete all search steps... | delete individual search steps...

Enter your search term(s): Search tips ☐ Thesaurus mapping

whole document

Information added since: (YYYYMMDD) or: none

search

☐ Documents with images

Select special search terms from the following list(s):

- Publication year 1950-
- Publication year 1898-1949
- Inspec thesaurus - browse headings A-G
- Inspec thesaurus - browse headings H-Q
- Inspec thesaurus - browse headings R-Z

10\763,732

Dial g DataStar

options logoff feedback help

databases search page titles

Document

Select the documents you wish to save or order by clicking the box next to the document, or click the link above the document to order directly.

save locally as: PDF document search strategy: do not include the search strategy

order

☑ Select All

1 A page fault equation for modeling the effect of memory size.
2 A self-tuning page cleaner for DB2.
3 VLRU: buffer management in client-server systems.
4 Integrating reliable memory in databases.
5 An analytical study of object identifier indexing.
6 Integrating reliable memory in databases.
7 Multimedia support for databases.
8 Holding a page: enhanced page level access control for database system
9 ARIES/CSA: a method for database recovery in client-server architec
10 Data base recovery in shared disks and client-server architectures.

☑ **document 1 of 10** Order Document

Inspec - 1898 to date (INZZ)

Accession number & update
0008805190 20060313.

Title
A **page** fault equation for modeling the effect of memory size.

Source
Performance Evaluation, {Perform-Eval-Netherlands}, Feb. 2006, vol. 63, no. 2, p. 99-130, 48 refs, CODEN: PEEVD9, ISSN: 0166-5316.
Publisher: Elsevier, Netherlands.

Author(s)
Tay-Y-C, Min-Zou.

Author affiliation
Tay, Y.C., Min Zou, Nat. Univ. of Singapore, Singapore.

Abstract
Modeling the effect of memory size on **page** faults is very difficult, because they are the result of the interaction between process reference behavior and **page** replacement policy. Moreover, a change in memory size will alter the timing and pattern of references in a multiprogramming mix. This paper presents an equation to model how memory size affects **page** faults. The equation is derived with the help of a conjectured invariant on the interaction between reference behavior and replacement policy. The **page** fault equation is validated in several experiments with real applications and systems. The equation can be used for energy conservation and capacity planning, characterizing the memory requirement of software and managing memory allocation, modeling of processor **cache** misses and **database buffer** management. (All rights reserved Elsevier).

Descriptors
CACHE-STORAGE; ENERGY-CONSERVATION; PAGED-STORAGE; RANDOM-ACCESS-

STORAGE; RESOURCE-ALLOCATION; STORAGE-ALLOCATION.

Classification codes

C6120 File-organisation*;
C6150N Distributed-systems-software.

Keywords

page-fault-equation; process-reference-behavior; **page-replacement-** policy; multiprogramming; memory-size; energy-conservation; capacity-planning; memory-requirement; memory-allocation; **database-buffer-** management; power-control; **processor-cache-design-modelling.**

Treatment codes

P Practical.

Language

English.

Publication type

Journal-paper.

Availability

SICI: 0166-5316(200602)63:2L.99:PFEM; 1-9.
Publisher identity number: S0166-5316(05)00018-0.

Digital object identifier

10.1016/j.peva.2005.01.007.

Publication year

2006.

Publication date

20060200.

Edition

2006010.

Copyright statement

Copyright 2006 IEE.

---

document 2 of 10 Order Document

Inspec - 1898 to date (INZZ)

Accession number & update

0007597206 20051201.

Title

A self-tuning **page** cleaner for DB2.

Conference information

Proceedings 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems. MASCOTS 2002, Fort Worth, TX, USA, 11-16 Oct. 2002.
Sponsor(s): IEEE Computer. Soc. Tech. Committee on Computer Architecture; IEEE Comput. Soc. Tech. Committee on Simulation; ACM SIGSIM; ACM SIGARCH.

Source

Proceedings 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems. MASCOTS 2002, 2002, p. 81-9, 8 refs, pp. xvi+521, ISBN: 0-7695-1840-0.
Publisher: IEEE Comput. Soc, Los Alamitos, CA, USA.

Author(s)

Wenguang-Wang, Bunt-R.
Editor(s): Boukreche-A, Das-S-K, Majumdar-S.

Author affiliation

Wenguang Wang, Bunt, R., Dept. of Comput. Sci., Saskatchewan Univ., Saskatoon, Sask., Canada.

Abstract

The **buffer** pool in a DBMS is used to **cache** the disk **pages** of the **database.** Because typical **database** workloads are I/O-bound, the effectiveness of the **buffer** pool management algorithm is a crucial factor in the performance of the DBMS. In IBM's DB2 **buffer** pool, the **page** cleaning algorithm

is used to write changed **pages** to disks before they are selected for replacement. We conducted a detailed study of **page** cleaning in DB2 version 7.1.0 for Windows by both trace-driven simulation and measurements. Our results show that system throughput can be increased by 19% when the **page** cleaning algorithm is carefully tuned. In practice, however the manual tuning of this algorithm is difficult. A self-tuning algorithm for **page** cleaning is proposed posed in this paper to automate this tuning task. Simulation results show that the self-tuning algorithm can achieve performance comparable to the best manually tuned system.

Descriptors

CACHE-STORAGE; DATABASE-MANAGEMENT-SYSTEMS.

Classification codes

C6160 Database-management-systems-DBMS*;
C6120 File-organisation.

Keywords

self-tuning-page-cleaner; IBM; **DB2-buffer-pool;** DBMS; **disk-pages- cache;** I/O-bound-workloads; **buffer-pool-management-algorithm; page-** cleaning-algorithm; DB2-version-7.1.0; Windows; trace-driven-simulation; system-throughput; manual-tuning; self-tuning-algorithm; simulation-results.

Treatment codes

A Application;
P Practical;
X Experimental.

Language

English.

Publication type

Conference-proceedings.

Availability

CCCC: 1526-7539/02/$17.00.

Digital object identifier

10.1109/MASCOT.2002.1167063.

Publication year

2002.

Publication date

20020000.

Edition

2003016.

Copyright statement

---

document 3 of 10 Order Document

Inspec - 1898 to date (INZZ)

Accession number & update

0006720999 20051201.

Title

VLRU: **buffer** management in client-server systems.

Source

IEICE Transactions on Information and Systems, {IEICE-Trans-Inf-Syst-Japan}, June 2000, vol. E83-D, no. 6, p. 1245-54, 22 refs, CODEN: ITISEF, ISSN: 0916-8532.
Publisher: Inst. Electron. Inf. & Commun. Eng, Japan.

Author(s)

Sung-Jin-Lee, Chin-Wan-Chung.

Author affiliation

Sung-Jin Lee, Dept. of Inf. & Commun. Eng., Korea Adv. Inst. of Sci. & Technol., Seoul, South Korea.

Abstract

In a client-server system, when LRU or its variant **buffer** replacement strategy is used on both the client and the server, the **cache** performance on the server side is very poor mainly because of **pages** duplicated in both systems. This paper introduces a server **buffer** replacement strategy which uses a replaced **page-id** rather than a request **page-id,** for the primary information for its operations. The importance of the corresponding **pages** in the server **cache** is decided according to the replaced **page-ids** that are delivered from clients to the server, so that locations of the **pages** are altered. Consequently, if a client uses LRU as its **buffer** replacement strategy, then the server **cache** is seen by the client as a long virtual client LRU **cache** extended to the server. Since the replaced **page-id** is only sent to the server by piggybacking whenever a new **page** fetch request is sent, the operation to deliver the replaced **page-id** is simple and induces a minimal overhead. We show that the proposed strategy reveals good performance characteristics in diverse situations, such as single and multiple clients, as well as with various access patterns.

Descriptors
CACHE-STORAGE; CLIENT-SERVER-SYSTEMS; DATABASE-MANAGEMENT-SYSTEMS; PAGED-STORAGE.

Classification codes
C6120 File-organisation*;
C6150N Distributed-systems-software;
C6160 Database-management-systems-DBMS.

Keywords
client-server-system; VLRU; **buffer-management; buffer-replacement-** strategy; **cache-performance; server-buffer-replacement-strategy; replaced-page-id; server-cache; pages; long-virtual-client-LRU-cache;** piggybacking; **page-fetch-request;** multiple-clients; single-clients; access-patterns.

Treatment codes
P Practical.

Language
English.

Publication type
Journal-paper.

Availability
SICI: 0916-8532(200006)E83D:6L.1245:VBMC; 1-Z.

Publication year
2000.

Publication date
20000600.

Edition
2000039.

Copyright statement
Copyright 2000 IEE.

document 4 of 10 Order Document

Inspec - 1898 to date (INZZ)

Accession number & update
0006225485 20051201.

Title
Integrating reliable memory in **databases.**

Conference information
Proceedings of VLDB 97: 23rd International Conference on Very Large **Databases,** Athens, Greece, 26-29 Aug. 1997.

Source
Proceedings of the Twenty-Third International Conference on Very Large **Databases,** 1997, p. 76-85, 41 refs, pp. xvi+599, ISBN: 1-55860-470-7.

Publisher: Morgan Kaufmann Publishers, San Francisco, CA, USA.

Author(s)

Wee-Teck-Ng, Chen-P-M.

Editor(s): Jarke-M, Carey-M, Dittrich-K-R, Lockovsky-F, Loucopoulos-P, Jeusfeld-M-A.

Author affiliation

Wee Teck Ng, Chen, P.M., Dept. of Electr. Eng. & Comput. Sci., Michigan Univ., Ann Arbor, MI, USA.

Abstract

Recent results in the Rio project at the University of Michigan show that it is possible to create an area of main memory that is as safe as disk from operating system crashes. This paper explores how to integrate the reliable memory provided by the Rio file **cache** into a **database** system. We propose three designs for integrating reliable memory into **databases:** non-persistent **database buffer cache,** persistent **database buffer cache,** and persistent **database buffer cache** with protection. Non-persistent **buffer caches** use an I/O interface to reliable memory and require the fewest modifications to existing **databases.** However, they waste memory capacity and bandwidth due to double buffering. Persistent **buffer caches** use a memory interface to reliable memory by mapping it into the **database** address space. This places reliable memory under complete **database** control and eliminates double buffering, but it may expose the **buffer cache** to **database** errors. Our third design reduces this exposure by write protecting the **buffer pages.** Extensive fault tests show that mapping reliable memory into the **database** address space does not significantly hurt reliability. This is because wild stores rarely touch dirty, committed **pages** written by previous transactions. As a result, we believe that **databases** should use a memory interface to reliable memory.

Descriptors

CACHE-STORAGE; DATABASE-MANAGEMENT-SYSTEMS; FAULT-TOLERANT-COMPUTING.

Classification codes

C6160 Database-management-systems-DBMS*;
C6120 File-organisation;
C5470 Performance-evaluation-and-testing.

Keywords

reliable-memory; **databases; Rio-file-cache; database-system; database- buffer-cache; persistent-database-buffer-cache; persistent-database- buffer-cache-with-protection; buffer-cache; database-errors.**

Treatment codes

P Practical.

Language

English.

Publication type

Conference-proceedings.

Publication year

1997.

Publication date

19970000.

Edition

1999016.

Copyright statement

Copyright 1999 IEE.

---

document 5 of 10 Order Document

Inspec - 1898 to date (INZZ)

Accession number & update

0006147207 20051201.

Title

An analytical study of object identifier indexing.

Conference information

Database and Expert Systems Applications. 9th International Conference, DEXA '98. Proceedings, Vienna, Austria, 24-28 Aug. 1998.
Sponsor(s): Univ. Vienna; Faculty of Law; DEXA-Assoc; OCG (Austrian Comput. Soc.); GI (Gesellschaft fur Informatik); et al.

Source

Database and Expert Systems Applications. 9th International Conference, DEXA'98. Proceedings, 1998, p. 38-49, 14 refs, pp. xvi +905, ISBN: 3-540-64950-6.
Publisher: Springer-Verlag, Berlin, Germany.

Author(s)

Norvag-K, Bratbergsengen-K.
Editor(s): Quirchmayr-G, Schweighofer-E, Bench-Capon-T-J-M.

Author affiliation

Norvag, K., Bratbergsengen, K., Dept. of Comput. & Inf. Sci., Norwegian Univ. of Sci. & Technol., Trondheim, Norway.

Abstract

To avoid OID index retrieval becoming a bottleneck, efficient buffering strategies are needed to minimize the number of disk accesses. In this paper, we develop analytical cost models which we use to find optimal sizes of the index **page buffer** and the index entry **cache,** for different memory sizes, index sizes, and access patterns. Because existing **buffer** hit estimation models are not applicable for index **page** buffering in the case of tree based indexes, we have also developed an analytical model for index **page buffer** performance. The cost gain from using the results in this paper is typically in the order of 200-300%. Thus, the results should be of valuable use in optimizers and tools for configuration and tuning of object-oriented **database** systems.

Descriptors

CACHE-STORAGE; DATABASE-INDEXING; DATABASE-THEORY; OBJECT-ORIENTED-DATABASES; QUERY-PROCESSING; TREE-DATA-STRUCTURES.

Classification codes

C6160J Object-oriented-databases*;
C6120 File-organisation;
C4250 Database-theory.

Keywords

object-identifier-indexing; OID-index-retrieval; efficient-buffering-strategies; minimized-disk-accesses; analytical-cost-models; optimal- **index-page-buffer-size; optimal-index-entry-cache-size;** memory-sizes; index-sizes; access-patterns; **buffer-hit-estimation-models;** tree-based-indexes; **index-page-buffer-performance;** cost-gain; optimizers; tools; configuration; tuning; **object-oriented-database-systems.**

Treatment codes

T Theoretical-or-mathematical.

Language

English.

Publication type

Conference-proceedings.

Publication year

1998.

Publication date

19980000.

Edition

1999004.

Copyright statement

Copyright 1999 IEE.

document 6 of 10 Order Document

Inspec - 1898 to date (INZZ)

Accession number & update
0006101585 20051201.

Title
Integrating reliable memory in **databases.**

Source
VLDB Journal, {VLDB-J-Germany}, Aug. 1998, vol. 7, no. 3, p. 194-204, 48 refs, CODEN: VLDBFR, ISSN: 1066-8888.
Publisher: Springer-Verlag, Germany.

Author(s)
Wee-Teck-Ng, Chen-P-M.

Author affiliation
Wee Teck Ng, Chen, P.M., Div. of Comput. Sci. & Eng., Michigan Univ., Ann Arbor, MI, USA.

Abstract
Results in the Rio project at the University of Michigan show that it is possible to create an area of main memory that is as safe as disk from operating system crashes. This paper explores how to integrate the reliable memory provided by the Rio file **cache** into a **database** system. Prior studies have analyzed the performance benefits of reliable memory; we focus instead on how different designs affect reliability. We propose three designs for integrating reliable memory into **databases:** non-persistent **database buffer cache,** persistent **database buffer cache,** and persistent **database buffer cache** with protection. Non-persistent **buffer caches** use an I/O interface to reliable memory and require the fewest modifications to existing **databases.** However, they waste memory capacity and bandwidth due to double buffering. Persistent **buffer caches** use a memory interface to reliable memory by mapping it into the **database** address space. This places reliable memory under complete **database** control and eliminates double buffering, but it may expose the **buffer cache** to **database** errors. Our third design reduces this exposure by write protecting the **buffer pages.** Extensive fault tests show that mapping reliable memory into the **database** address space does not significantly hurt reliability. This is because wild stores rarely touch dirty, committed **pages** written by previous transactions. As a result, we believe that **databases** should use a memory interface to reliable memory.

Descriptors
CACHE-STORAGE; OPERATING-SYSTEMS-COMPUTERS; SOFTWARE-PERFORMANCE-EVALUATION; SOFTWARE-RELIABILITY; VERY-LARGE-DATABASES.

Classification codes
C6160Z Other-DBMS*;
C6110B Software-engineering-techniques;
C6120 File-organisation.

Keywords
reliable-memory; **databases;** Rio-project; University-of-Michigan; main-memory; operating-system-crash; **file-cache;** performance-benefits; **nonpersistent-database-buffer-cache; persistent-database-buffer-cache;** input-output-interface; memory-capacity; double-buffering; memory-interface.

Treatment codes
P Practical.

Language
English.

Publication type
Journal-paper.

Availability
SICI: 1066-8888(199808)7:3L.194:IRMD; 1-L.
CCCC: 1066-8888/98/$2.00+0.20.

Publication year
1998.

Publication date
19980800.

Edition
1998048.

Copyright statement
Copyright 1998 IEE.

document 7 of 10 Order Document
Inspec - 1898 to date (INZZ)

Accession number & update
0005871118 20051201.
Title
Multimedia support for **databases.**
Conference information
Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of **Database** Systems, PODS 1997, Tucson, AZ, USA, 12-14 May 1997.
Sponsor(s): ACM.
Source
Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of **Database** Systems, PODS 1997, 1997, p. 1-11, 38 refs, pp. viii+268, ISBN: 0-89791-910-6.
Publisher: ACM, New York, NY, USA.
Author(s)
Ozden-B, Rastogi-R, Silberschatz-A.
Author affiliation
Ozden, B., Rastogi, R., Silberschatz, A., AT&T Bell Labs., Murray Hill, NJ, USA.
Abstract
Next-generation **database** systems will need to provide support for both textual data and other types of multimedia data (e.g. images, video, audio). These two types of data differ in their characteristics, and hence require different techniques for their organization and management. For example, continuous-media data (e.g. video, audio) requires a guaranteed transfer rate. In this paper, we provide an overview of (1) how **database** systems can be architectured to support multimedia data, and (2) the main challenges in devising new algorithms to manage multimedia data. In order to provide rate guarantees for continuous-media data, an admission control scheme must be employed that determines, for each client, whether there are sufficient resources available to service that client. To maximize the number of clients that can be admitted concurrently, the various system resources must be allocated and scheduled carefully. In terms of disks, we use algorithms for retrieving/storing data from/to disks that reduce the seek latency time and eliminate rotational delay, thereby providing high throughput. In terms of main memory, we use **buffer** management schemes that exploit the sequential access patterns for continuous-media data, thereby resulting in efficient replacement of **buffer pages** from the **cache.** In addition to discussing resource scheduling, we also present schemes for the storage layout of data on disks and schemes that provide fault tolerance by ensuring uninterrupted service in the presence of disk failures.
Descriptors
CACHE-STORAGE; CLIENT-SERVER-SYSTEMS; CONCURRENCY-CONTROL; DISTRIBUTED-DATABASES; FAULT-TOLERANT-COMPUTING; INFORMATION-STORAGE; MULTIMEDIA-COMPUTING; PAGED-STORAGE; RESOURCE-ALLOCATION; SCHEDULING.
Classification codes
C6160B Distributed-databases*;
C6120 File-organisation;
C6130M Multimedia;
C6160S Spatial-and-pictorial-databases.
Keywords
multimedia-support; **database-systems;** textual-data; continuous-media-data; guaranteed-transfer-rate; **database-system-architecture;** multimedia-data-management-algorithms; admission-control-scheme; resource-determination; concurrent-client-number-maximization; system-resource-allocation; resource-scheduling; disk-storage; data-retrieval; seek-latency-time; rotational-delay; throughput; main-memory; **buffer-management-schemes;** sequential-access-patterns; **buffer- page-replacement; cache;** data-storage-layout; fault-tolerance; uninterrupted-service; disk-failures.
Treatment codes
P Practical.
Language
English.
Publication type

Conference-proceedings.

Availability
CCCC: 0 89791 910 6/97/05..$3.50.

Publication year
1997.

Publication date
19970000.

Edition
1998012.

Copyright statement
Copyright 1998 IEE.

☑ **document 8 of 10** Order Document

Inspec - 1898 to date (INZZ)

Accession number & update
0005219986 20051201.

Title
Holding a **page:** enhanced **page** level access control for **database** systems.

Conference information
ADC'96. Seventh Australasian **Database** Conference, Melbourne, Vic., Australia, 29-30 Jan. 1996.

Source
Australian Computer Science Communications, {Aust-Comput-Sci-Commun-Australia}, 1996, vol. 18, no. 2, p. 38-47, 16 refs, CODEN: ACSCDD, ISSN: 0157-3055.
Publisher: James Cook Univ, Australia.

Author(s)
Harris-E-P, Ramamohanarao-K.

Author affiliation
Harris, E.P., Ramamohanarao, K., Dept. of Comput. Sci., Melbourne Univ., Parkville, Vic., Australia.

Abstract
A new scheme for physically accessing tuples by multiple transactions is described for **database** systems using a shared-memory **buffer cache.** Using this scheme, a transaction must first obtain logical permission to access a tuple, e.g. by taking a lock on the tuple. Once logical permission has been granted, a **page-level** latch is taken for the initial physical access to the tuple. However, by the introduction of a new primitive (<b>hold</b>) into the latching system, which ensures that the tuple is not moved, a latch on the **page** is not required to subsequently read the tuple. This brings the physical level of concurrency closer to the logical level of concurrency. Our scheme is compared with other schemes which can be used for access control.

Descriptors
AUTHORISATION; CACHE-STORAGE; CONCURRENCY-CONTROL; DATABASE-MANAGEMENT-SYSTEMS; PAGED-STORAGE; SHARED-MEMORY-SYSTEMS; TRANSACTION-PROCESSING.

Classification codes
C6120 File-organisation*;
C6130S Data-security;
C6160 Database-management-systems-DBMS.

Keywords
page-holding; page-level-access-control; database-systems; physical- tuple-access-scheme; multiple-transactions; **shared-memory-buffer- cache;** transaction; logical-permission; tuple-lock; **page-level-latch;** hold-primitive; concurrency.

Treatment codes
P Practical.

Language
English.

Publication type
Conference-proceedings; Journal-paper.
Availability
SICI: 0157-3055(1996)18:2L.38:HPEP; 1-0.
Publication year
1996.
Publication date
19960000.
Edition
1996012.
Copyright statement
Copyright 1996 IEE.

☑ **document 9 of 10** Order Document
Inspec - 1898 to date (INZZ)

Accession number & update
0004726613 20051201.
Title
ARIES/CSA: a method for **database** recovery in client-server architectures.
Conference information
1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, MN, USA, 24-27 May 1994.
Sponsor(s): ACM.
Source
SIGMOD Record, {SIGMOD-Rec-USA}, June 1994, vol. 23, no. 2, p. 55-66, 30 refs, CODEN: SRECD8, ISSN: 0163-5808, USA.
Author(s)
Mohan-C, Narang-I.
Author affiliation
Mohan, C., Narang, I., Data Base Technol. Inst., IBM Almaden Res. Center, San Jose, CA, USA.
Abstract
Presents an algorithm called ARIES/CSA (Algorithm for Recovery and Isolation Exploiting Semantics for Client-Server Architectures) for performing recovery correctly in client-server architectures where the server manages the disk version of the **database.** The clients, after obtaining **database pages** from the server, **cache** them in their **buffer** pools. Clients perform their updates on the **cached pages** and produce log records. The log records are buffered locally in virtual storage and later sent to the single log at the server. ARIES/CSA supports write-ahead logging, fine-granularity (e.g. record) locking, partial rollbacks and flexible **buffer** management policies like `steal' and `no-force'. It does not require that the clocks on the clients and the server be synchronized. Checkpointing by the server and the clients allows for flexible and easier recovery.
Descriptors
BUFFER-STORAGE; CLOCKS; DISTRIBUTED-DATABASES; SYSTEM-RECOVERY; VIRTUAL-STORAGE.
Classification codes
C6160B Distributed-databases*.
Keywords
ARIES/CSA; **database-recovery;** client-server-architectures; isolation; semantics; **database-disk-version; database-pages;** locally-buffered-log-records; **buffer-pools;** updates; **cached-pages;** virtual-storage; write-ahead-logging; fine-granularity-locking; record-locking; partial-rollbacks; **flexible-buffer-management-policies;** steal; no-force; clocks; checkpointing.
Treatment codes
P Practical.
Language

English.
Publication type
Conference-proceedings; Journal-paper.
Availability
CCCC: 0163-5808/94/0005$3.50.
Publication year
1994.
Publication date
19940600.
Edition
1994030.
Copyright statement
Copyright 1994 IEE.

☑ **document 10 of 10** Order Document
Inspec - 1898 to date (INZZ)

Accession number & update
0004468185 20051201.
Title
Data base recovery in shared disks and client-server architectures.
Conference information
Proceedings of the 12th International Conference on Distributed Computing Systems (Cat No.92CH3175-7), Yokohama, Japan, 9-12 June 1992.
Sponsor(s): IEEE; Inf. Process. Soc. Japan.
Source
Proceedings of the 12th International Conference on Distributed Computing Systems (Cat No.92CH3175-7), 1992, p. 310-17, 32 refs, pp. xxii+725, ISBN: 0-8186-2865-0.
Publisher: IEEE Comput. Soc. Press, Los Alamitos, CA, USA.
Author(s)
Mohan-C, Narang-I.
Author affiliation
Mohan, C., Narang, I., IBM Almaden Res. Center, San Jose, CA, USA.
Abstract
Solutions to the problem of performing recovery correctly in shared-disks (SD) and client-server (CS) architectures are presented. In SD, all the disks containing the data bases are shared among multiple instances of the **database** management system (DBMS). In CS, the server manages the disk version of the data base. The clients, after obtaining **database pages** from the server, **cache** them in their **buffer** pools. Clients perform their updates on the **cached pages** and produce log records. In write-ahead logging (WAL) systems, a monotonically increasing value called the log sequence number (LSN) is associated with each log record. Every **database page** contains the LSN of the log record describing the most recent update to that **page.** This is required for proper recovery after a system failure. A technique for generating monotonically increasing LSNs in SD and CS architectures without using synchronized clocks is presented.
Descriptors
DATABASE-MANAGEMENT-SYSTEMS; PERFORMANCE-EVALUATION.
Classification codes
C6160 Database-management-systems-DBMS*;
C5470 Performance-evaluation-and-testing.
Keywords
database-recovery; shared-disks; client-server-architectures; data- bases; multiple-instances; **database-management-system; database-pages; buffer-pools; cached-pages;** log-records; write-ahead-logging; log- sequence-number.
Treatment codes

P Practical.

Language
English.

Publication type
Conference-proceedings.

Availability
CCCC: 0 8186 2865 0/92/$3.00.

Digital object identifier
10.1109/ICDCS.1992.235026.

Publication year
1992.

Publication date
19920000.

Edition
1993032.

Copyright statement
Copyright 1993 IEE.

save locally as: PDF document search strategy: do not include the search strategy

order

Document

Select the documents you wish to save or order by clicking the box next to the document, or click the link above the document to order directly.

next documents

save locally as: PDF document search strategy: do not include the search strategy

order

☑ **document 1 of 2** Order Document

Inspec - 1898 to date (INZZ)

Accession number & update
0008949474 20060625.

Title
Improving trace **cache** processor performance by trace **cache** hierarchy and path-based trace prefetch.

Source
Chinese Journal of Electronics, {Chin-J-Electron-China}, April 2006, vol. 15, no. 2, p. 231-6, 10 refs, CODEN: CHJEEW, ISSN: 1022-4653.
Publisher: Chinese Inst. Electron, China.

Author(s)
Wang-Kaifeng, Ji-Zhenzhou, Hu-Mingzeng.

Author affiliation
Wang Kaifeng, Ji Zhenzhou, Hu Mingzeng, Sch. of Comput. Sci. & Technol., Harbin Inst. of Technol., China.

Abstract
The performance of trace **cache** processor rests with trace **cache** efficiency to a great **extent.** Higher trace **cache** miss rate will reduce performance significantly because only a low fetch-bandwidth can be maintained by conventional instruction **cache.** Unfortunately, with the ever increasing conventional application scale, higher trace **cache** miss rate is inevitable for the relative small capacity of trace **cache,** which will become the bottleneck of performance improvement. In this paper, we proposed trace **cache** hierarchy to remedy the limited capacity of 1-level trace **cache.** 2-level trace **cache** is incorporated in trace processors. But the simulation results show that only augmenting 2-level trace **cache** can not bring significant performance improvement for the long access latency. So we propose a path-based trace prefetch mechanism to reduce the latency of 2-level trace **cache** access further. Path-based trace prefetch mechanism is developed on top of next N trace prediction mechanism. By predicting the next N trace from current and prefetching it into trace prefetch **buffer** from 2-level trace **cache,** the access latency of 2-level trace **cache** can be reduced. The simulation results show that augmenting an 8K-Entry, eight-way 2-level trace **cache,** an 16-Entry trace prefetch **buffer** and prefetch distance set to 3, the average IPC improvement is 12.0% for eight SPECint95 benchmarks.

Descriptors
CACHE-STORAGE; MICROPROCESSOR-CHIPS; STORAGE-MANAGEMENT.

Classification codes
B1265F Microprocessors-and-microcomputers*;
B1265D Memory-circuits;

C5130 Microprocessor-chips*;
C5320G Semiconductor-storage;
C6120 File-organisation.

Keywords
trace-cache-processor-performance; trace-cache-hierarchy; path-based- trace-prefetch; **trace-cache-miss-rate; instruction-cache;** trace-prediction-mechanism; **trace-prefetch-buffer;** access-latency.

Treatment codes
P Practical.

Language
English.

Publication type
Journal-paper.

Availability
SICI: 1022-4653(200604)15:2L.231:ITCP; 1-5.

Publication year
2006.

Publication date
20060400.

Edition
2006025.

Copyright statement
Copyright 2006 The Institution of Engineering and Technology.

next documents
save locally as: PDF document search strategy: do not include the search strategy
order

Top - News & FAQS - Dialog

Dialog DataStar

options logoff feedback help

databases search page titles

Document

Select the documents you wish to save or order by clicking the box next to the document, or click the link above the document to order directly.

previous documents

save locally as: PDF document search strategy: do not include the search strategy

order

☑ **document 2 of 2** Order Document

Inspec - 1898 to date (INZZ)

Accession number & update
0007685765 20051201.

Title
SANtopia: shared-disk file system for storage cluster.

Conference information
PDCS 2002: 14th IASTED International Conference on Parallel and Distributed Computing and Systems, Cambridge, MA, USA, 4-6 Nov. 2002.
Sponsor(s): IASTED.

Source
Proceedings of the 14th IASTED International Conference Parallel and Distributed Computing and Systems, 2002, p. 464-9, 13 refs, pp. vi +860, ISBN: 0-88986-366-0.
Publisher: ACTA Press, Anaheim, CA, USA.

Author(s)
Yong-Ju-Lee, Choo-Seo-Park, Gyoung-Bae-Kim, Kee-Wok-Rim, Bum-Joo-Shin.

Author affiliation
Yong-Ju Lee, Choo-Seo Park, Gyoung-Bae Kim, Kee-Wok Rim, Dept. of Comput. Syst., Electron. & Telecommun. Res. Inst., Daejeon, South Korea.

Abstract
There have been large storage demands for manipulating multimedia data such as images and video. To solve tremendous storage demands, one major research is the SAN (storage area network) that provides local file requests directly from shared disk storage and also eliminates server bottlenecks to performance and availability. The SAN also improves network latency and bandwidth through a new channel interface like FC (fibre channel). The FC is capable of maintaining several simultaneous gigabit data transfer, but to make use of an efficient storage network like SAN, a traditional file system is not adaptable in terms of scalability, availability and consistency issues. We propose a he new shared-disk file system, the so-called SANtopia file system, for shared disk storage that manipulates large-scale inode objects and provides key cluster enabling technology for Linux, helping to bring the scalability, availability and load balancing benefits of clustering to Linux. We describe the architecture and design issues of a shared-disk file system for shared disk storage and provide the efficient bitmap, **extent-based** semi-flat structure and two-phase directory structure using extendible hashing. We also present a **cache** coherence protocol using a **buffer** forwarding scheme to maintain efficient metadata consistency. We evaluate the performance in terms of average response time and I/O rate.

Descriptors
DISTRIBUTED-DATABASES; FILE-SERVERS; META-DATA; SHARED-MEMORY-

SYSTEMS;
STORAGE-MANAGEMENT.

Classification codes
C6120 File-organisation*;
C6160B Distributed-databases.

Keywords
SANtopia; shared-disk-file-system; storage-cluster; multimedia; data-manipulation; storage-area-network; shared-disk-storage; network-server; network-latency; network-bandwidth; FC; fibre-channel; gigabit-data-transfer; storage-network; Linux; load-balancing; two-phase-directory-structure; extendible-hashing; **cache-coherence-** protocol; **buffer-forwarding-scheme;** metadata-consistency; I/O-rate.

Treatment codes
P Practical.

Language
English.

Publication type
Conference-proceedings.

Publication year
2002.

Publication date
20020000.

Edition
2003027.

Copyright statement
Copyright 2003 IEE.

previous documents
save locally as: PDF document search strategy: do not include the search strategy
order

WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Friday, August 04, 2006

Hide?	Set Name	Query	Hit Count
		DB=PGPB,USPT,USOC; PLUR=NO; OP=OR	
☐	L55	(L53 or L54) and (search$ or quer$ or request$ or inquir$ or enquir$ or question)	85
☐	L54	L52 and ((extent or extents) near page$)	2
☐	L53	L52 and ((database$ or (data adj1 base$)) with table$)	86
☐	L52	(L50 or L51) and (buffer adj1 cach$)	235
☐	L51	(707/200 |707/201 |707/202 |707/203 |707/204 |707/205).ccls.	7067
☐	L50	(707/2 |707/3 |707/4).ccls.	9745
☐	L49	L48 and engine$	28
☐	L48	L47 and (search$ or quer$ or request$ or inquir$ or enquir$ or question)	28
☐	L47	L46 and (buffer adj1 cach$)	28
☐	L46	(database adj1 engine$)	2373
☐	L45	L44 and engine$	1
☐	L44	20030041214.pn.	1
☐	L43	L42 and (engine$ near (search$ or quer$ or request$ or inquir$ or enquir$ or question$))	33
☐	L42	(buffer adj1 cach$)	2240
☐	L41	L40 and (buffer near cach$)	18
☐	L40	L39 and (engine$ near (search$ or quer$ or request$ or inquir$ or enquir$ or question$))	4198
☐	L39	((database$ or (data adj1 base$)) with table$)	53497
☐	L38	L37 and ((database$ or (data adj1 base$)) with table$)	16
☐	L37	L36 and cach$	97
☐	L36	L35 and (engine$ near (search$ or quer$ or request$ or inquir$ or enquir$ or question$))	203
☐	L35	((extent or extents) with (page or pages))	3078
☐	L34	L33 and (extent or extents)	2
☐	L33	L31 and (buffer with cach$)	15
☐	L32	L31 and (buffer near cach$)	1
☐	L31	L30 and (engine$ near (search$ or quer$ or request$ or inquir$ or enquir$ or question$))	1486
☐	L30	(international adj1 business).asn.	63605
☐	L29	2002198872.pn.	0

10/763,752

☐	L28	2002198872.pn.	0
☐	L27	L26 and (engine$ near (search$ or quer$ or request$ or inquir$ or enquir$ or question$))	17
☐	L26	((extent or extents) near page$)	201
☐	L25	L16 and ((extent or extents) near page$)	2
☐	L24	L23 and page$	11
☐	L23	(L19 or L20) and ((database$ or (data adj1 base$)) with table$)	11
☐	L22	(L19 or L20) and (memory with (database$ or (data adj1 base$)) with table$)	0
☐	L21	L16 and (memory with (database$ or (data adj1 base$)) with table$)	2
☐	L20	L18 and (extent or extents).ab.	58
☐	L19	L18 and (extent or extents).ti.	1
☐	L18	(engine$ near (search$ or quer$ or request$ or inquir$ or enquir$ or question$))	19038
☐	L17	L16 and engine$	1
☐	L16	(buffer near cach$ near (extent or extents))	2
☐	L15	L5 and (search$ or request$ or inquir$ or enquir$ or question$ or quer$)	2
☐	L14	L12 and page$	1
☐	L13	(L8 or L9 or L10) and engine$	0
☐	L12	L11 and engine$	1
☐	L11	L1 and ((search$ or quer$ or request$ or inquir$ or enquir$ or question$) near (extent or extents))	5
☐	L10	L1 and ((extent or extents) same (buffer adj1 cach$))	2
☐	L9	L1 and ((extent or extents) with (buffer adj1 cach$))	2
☐	L8	L1 and ((extent or extents) near (buffer adj1 cach$))	1
☐	L7	L1 and ((search$ or quer$ or request$ or inquir$ or enquir$ or question$) with (extent or extents) with (buffer adj1 cach$))	0
☐	L6	L5 and engine$	1
☐	L5	L4 and (memory with (database$ or (data adj1 base$)) with table$)	2
☐	L4	L3 and (database$ or (data adj1 base$))	5
☐	L3	((buffer adj1 cache) with (extent or extents))	10
☐	L2	((buffer adj1 cache) near (extent or extents))	2

DB=USPT; PLUR=NO; OP=OR

(5317727 5812996 5822749 5758149 5794229 5794228 5655080 6374232
6973452 6101497 6442551 6021426 5903898 5956705 6457020 6470330
6243710 5668987 6073129 6105033 6122627 6134540 6226637 6226637
6285997 6341281 6470344 6477527 6574639 6801905 6889234 5832508
5737536 5826253 5850507 6182241 6898608 6957177 5201046 5511190
5742806 5918225 6289334 5884303 6654752 6961729 7010308 6389513
5787418 5842209).pn. (6078926 5426747 5940289 6125209 5581704 5615362
5706506 5802524 5832475 5941947 5944780 5974129 5999946 6009271
6070165 6122628 6128648 6134541 6185557 6279033 6282281 6360214
6381627 6401090 6411966 6438562 6449657 6466570 6487641 6532490

☐	L1	6539382 6601062 6604096 6694306 6694322 6732117 6741997 6763357 6898603 6950823 7062480 6728840 6912636 5799210 6230220 6591351 6760824 6904503 6347312 6748386).pn. (6014655 6115705 5210870 5237661 5454105 5530883 5537622 5537604 5537603 5548769 5590362 5619713 5745915 5758146 5778353 6457000 6691101 6754825 6836845 5418940 5802599 6049848 5907846 5010478 5283894 5542078 5781897 5918224 6115703 6928451 5561778 5579499 5590319 5594881 5600831 5701460 5737591 6134018 6253195 5506984 5694608 5893125 6138112 5317731 5495606 5546576 5560007 5596744 5634053 5664173).pn. (5666528 5724570 5412806 5603025 5692182 5692174 5727196 5787416 5845288 5870752 5894311 5903887 5937401 5950188 6006224 6012064 6044370 6076092 6081801 6212526 5201048 5265244 5329626 5367675 5423022 5504885 5574900 5596745 5615337 5619688 5627959 5632015 5694591 5701456 5701453 5749079 5761493 5761653 5774692 5794231 5797136 5799310 5806066 5826077 5835904 5842197 5842196 5852821 5878426 5897622).pn. (5905982 5918232 5924089 5930795 5930764 5937415 5943666 5953715 5956727 5960426 5966695 5974407 5974418 5987454 5991754 5995958 5995957 5995973 6006214 6009265 6009428 6012054 6016488 6023695 6023696 6026391 6044216 6047285 6047291 6078925 6081799 6085189 6088694 6092061 6105025 6108647 6108648 6112198 6125360 6134543 6134546 6138120 6148296 6192370 6199062 6199063 6212526 6249783 6249791 6263339).pn. (6272487 6282547 6298342 6324533 6353826 6356889 6363387 6381605 6421658 6427123 6430556 6453269 6460043 6470287 6470335 6477525 6477540 6496819 6502088 6507834 6516310 6549907 6557012 6560593 6571232 6581052 6581060 6584476 6598059 6615202 6615206 6618719 6631386 6633882 6636846 6691166 6697818 6708179 6708186 6714938 6732084 6735582 6735598 6741982 6748377 6757670 6799184 6801850 6807546).pn.	297

END OF SEARCH HISTORY